Robust Predictive Tracking Control for A Class of Nonlinear Systems

T.Q. Dinh[1,*], J. Marco[1], J.I. Yoon[2], K.K. Ahn[3]

[1] Warwick Manufacturing Group (WMG), University of Warwick, Coventry CV4 7AL, UK;
[2] Korea Construction Equipment Technology Institute, Jeollabuk-do Gunsan 573-540 Sandan-ro 36, Korea;
[3] School of Mechanical Engineering, University of Ulsan, Namgu Muger2dong, Ulsan 680-749, Korea;
* Correspondence: q.dinh@warwick.ac.uk; Tel.: +44-2476-574902

ABSTRACT

A robust predictive tracking control (RPTC) approach is developed in this paper to deal with a class of nonlinear SISO systems. To improve the control performance, the RPTC architecture mainly consists of a robust fuzzy PID (RFPID) –based control module and a robust PI grey model (RPIGM) –based prediction module. The RFPID functions as the main control unit to drive the system to desired goals. The control gains are online optimized by neural network-based fuzzy tuners. Meanwhile using grey and neural network theories, the RPIGM is designed with two tasks: to forecast the future system output which is fed to the RFPID to optimize the controller parameters ahead of time; and to estimate the impacts of noises and disturbances on the system performance in order to create properly a compensating control signal. Furthermore, a fuzzy grey cognitive map (FGCM) –based decision tool is built to regulate the RPIGM prediction step size to maximize the control efforts. Convergences of both the predictor and controller are theoretically guaranteed by Lyapunov stability conditions. The effectiveness of the proposed RPTC approach has been proved through real-time experiments on a nonlinear SISO system.

Keywords

Fuzzy, PID, neural network, grey predictor, fuzzy cognitive map, nonlinear system.

1. Introduction

Nowadays, automation in control has been applied more and more in the modern life. However, most of industrial machines are nonlinear systems with large uncertainties which cause challenges to design the controllers. Conventional PID controllers are commonly used in industry due to their simplicity, clear functionality and ease of implementation. However, this type of controllers may not perform well for nonlinear, complex and vague systems with uncertainties. And it has been found that fuzzy-logic-based PID controllers is one of potential solutions with better capabilities of handling the aforesaid systems [2]-[17].

Although fuzzy logic has a reputation of handling complicated control problems, typical fuzzy designs depend largely on experiences of experts [1]-[5]. Hence, these controllers cannot adapt for highly uncertain systems working in environments with large perturbations [9], [12]. There is no systematic method to design and examine the number of rules, as well as input space partitions and membership functions (MFs). As a result, other control techniques, such as robust control, intelligent theory and estimation methods [6]-[14], are needed to combine with the fuzzy PID to overcome this weakness. Nevertheless, most of the traditional control strategies adopted the previous state information as the input signal of the controllers to make the decisions. Subsequently, this type of control reflects only the current status and lacks adaptability.

As a recent trend to overcome this drawback, fuzzy PID combined with prediction theories could produce in advance the control action for the following step according to the predicted value of control error before it occurs [15], [16]. And the combination with neural technique and grey prediction is a feasible solution. Neural network is a universal algorithm which is able to approximate almost nonlinear functions [17]-[24] while the grey theory [25] is distinguished by its ability to deal with systems that have partially unknown parameters [15],[16],[26]-[35]. However, there are the shortcomings of the typical grey models such as grey sequence conditions and background series calculation which limit their applicability as well as prediction accuracy [34], [35]. Additionally, there is no constraint to guarantee the prediction stability of these developed models.

The aim of this paper is to develop a robust predictive tracking control (RPTC) approach to improve performances of SISO systems with large nonlinearities and uncertainties. The RPTC

architecture mainly consists of two modules: robust PI grey model (RPIGM) -based prediction module and robust fuzzy PID (RFPID) -based control module with the following contributions:

1) To deal with any signal with random distribution, the RPIGM is newly developed using a closed-loop control form in which the robust prediction performance is ensured by a PI-based neural network controller.
2) Outputs from the RPIGM module are fed to the RFPID control module to optimize its parameters and, used to compensate for the impacts of noises and disturbances on the overall system response.
3) The RFPID of which the control gains are regulated by fuzzy tuners is designed to drive the system to a desired goal. Based on the RPIGM outputs, the control parameters are optimized in advance by a neural network-based learning mechanism.
4) A fuzzy grey cognitive map (FGCM) –based decision tool is built and integrated to the RPIGM to regulate online the RPIGM prediction step size in order to maximize the control capability.
5) The robust performances of both the RFPID and RPIGM are guaranteed by the Lyapunov stability conditions.

As the result, the overall control performance with high accuracy, fast response and stability can be achieved. This paper is organized as: Section II shows the system description and the RPTC architecture. Section III presents the design of the RFPID control module while the design of the RPIGM prediction module is described in Section IV. Illustrative examples via real-time experiments are provided and discussed in Section V to verify the effectiveness of the proposed control methodology. Finally, concluding remarks are given in Section VI.

2. System description and RPTC design architecture

Without loss of generality, the RPTC control scheme is designed for an uncertain nonlinear system (P) with single-input-single-output (SISO) [12] as in Fig. 1. The proposed RPTC architecture with the two modules, RFPID and RPIGM, is employed to drive the system to follow a given reference (R) (the system response $y(t) \equiv y_t$ needs to reach to the desired level $y_r(t) \equiv y_{rt}$).

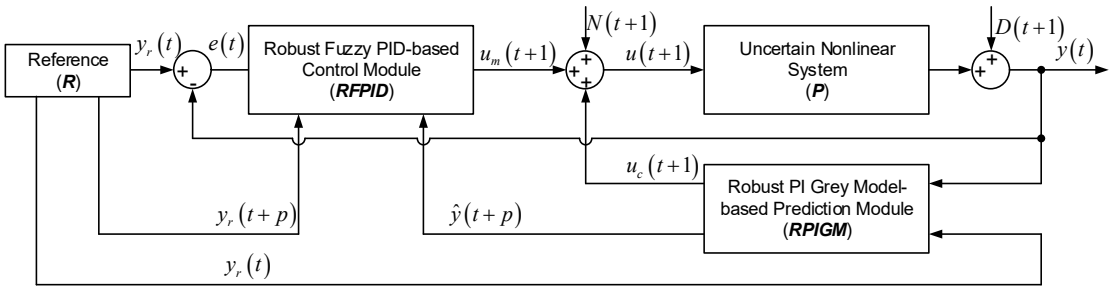


Fig. 1. Overall RPTC control architecture for a generic nonlinear system.

At step $(t+1)^{th}$ with the tracking error, $e(t) = y_r(t) - y(t)$, the RFPID generates a proper control action based on the PID algorithm, $u_{RFPID}(t+1) \equiv u_m(t+1) \equiv u_{m(t+1)}$. Meanwhile using the information of $y_r(t)$ and $y(t)$, the RPIGM estimates the system actuation p-step ahead of time, $\hat{y}(t+p) \equiv \hat{y}_{t+p}$. This estimated response is then employed with the p-step ahead desired set point, $y_r(t+p)$, to optimize robustly the RFPID parameters. Moreover, the RPIGM produces an additive control correction, $u_{RPIGP}(t+1) \equiv u_c(t+1) \equiv u_{c(t+1)}$, which is added to the main control signal $u_{m(t+1)}$, to compensate for system noises (N) and disturbances (D). Therefore, the system control input generated by the RPTC scheme is computed as

$$u_{RPTC}(t+1) \equiv u(t+1) = u_m(t+1) + u_c(t+1) \tag{1}$$

$$u_m(t+1) = K_P(t+1)e(t) + K_I(t+1)\int_0^t e(t)dt + K_D(t+1)\frac{de(t)}{dt} \tag{2}$$

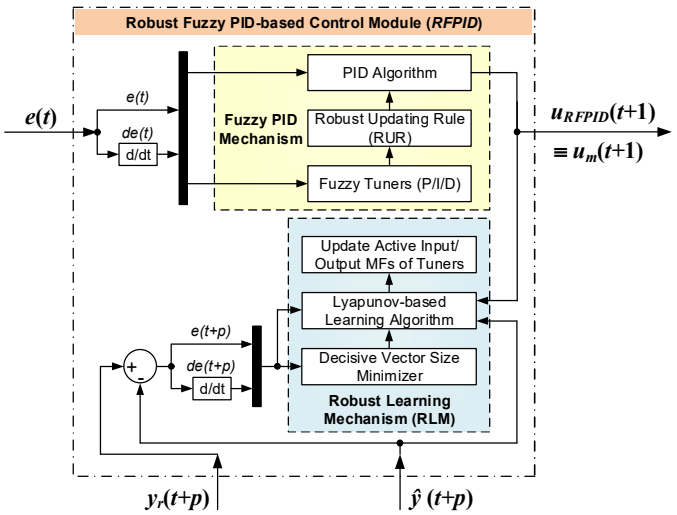
$$u_c(t+1) = K_c \times \hat{e}_{ND}^{(0)}(t+1) \tag{3}$$

where: $e(t)$ is the control error; $de(t)$ is the derivation of error $e(t)$; $\hat{e}_{ND}^{(0)}(t+1) = \hat{y}(t+1) - y(t+1)$ is the impact of noise and disturbance on the system response, $\hat{y}(t+1)$ is estimated by the RPIGM; $K_P(t+1)$, $K_I(t+1)$, and $K_D(t+1)$ are the dynamic proportional, integral, and derivative gains of the PID algorithm, respectively, regulated by fuzzy inferences; K_c is the fixed conversion factor.

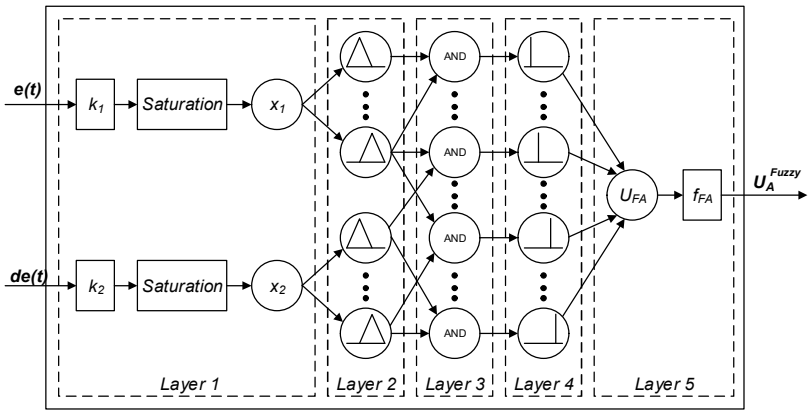
The detailed designs of the RFPID and RPIGM modules are introduced in the following sections.

3. Robust fuzzy PID-based control module

Structure of the RFPID control module (shown in Fig. 1) is described in Fig. 2a. This module includes two main blocks: a fuzzy PID mechanism, which is the combination of the PID algorithm and three fuzzy tuners to regulate the PID gains via a robust updating rule (RUR), to produce the control output, and a robust learning mechanism (RLM) to optimize parameters of the fuzzy tuners.



(a) Internal structure of RFPID



(b) Schematic diagram of a fuzzy tuner A (A representing P or I or D)

Fig. 2. RFPID control module.

3.1. Fuzzy PID mechanism

a) Fuzzy tuners

To minimize the tracking error, the PID gains, K_P, K_I and K_D, are online regulated using the three separate fuzzy tuners: fuzzy P, fuzzy I and fuzzy D, respectively, as the following:

$$K_A^{Fuzzy}(t+1) = K_{A0} + U_A^{Fuzzy}(t+1)\Delta K_A$$
$$U_A^{Fuzzy}(t+1) \in (0,1), \quad (A \text{ is } P, I \text{ or } D) \tag{4}$$

where $\Delta K_A = K_{A1} - K_{A0}$ is the allowable deviation of K_A; K_{A0}, K_{A1} are the minimum, maximum values of K_A, respectively; $U_A^{Fuzzy}(t+1)$ is the bounded parameter and derived from the fuzzy tuner P or I or D. Thus, one has $K_A(t+1) \subset \left[K_A^{\min}, K_A^{\max}\right]$.

Remark 1. For all the fuzzy tuners, triangle and singleton MFs are used to represent for partitions of fuzzy inputs and outputs, respectively. Fuzzy control is applied using local inferences. That means each rule is inferred and the inferring results of individual rules are then aggregated. Here, the most common inference using max-min method, which offers a computationally nice and expressive setting for constraint propagation, is selected. Finally, a defuzzification is needed to obtain a crisp output from the aggregated fuzzy result. The centroid defuzzification, which is widely used for fuzzy control problems needing crisp outputs, is chosen to construct the fuzzy tuners.

From (2), (4) and using Remark 1, each fuzzy tuner are designed with two inputs (as the most practical fuzzy PID type [12]) and one output as depicted in Fig. 2b. For the optimisation purpose, each tuner is structured in the network form with five layers. In the layer 1, the two inputs x_1 and x_2 are the same for both the tuners and derived as normal scales or absolute scales of the control error and its derivative, which are depended on the symmetric behaviour of the system. The range for each fuzzy input is correspondingly forced into range from -1 to 1 or from 0 to 1 by proper scaling factors (k_1 and k_2) chosen from the system specifications. These inputs are then converted into fuzzy values via the layer 2 using triangle MFs. Each MF of each input variable can be expressed in a general form as follows:

$$f_j(x_i) = \begin{cases} 1 + (x_i - a_{ij}) / b_{ji}^- & \text{if } (-b_{ij}^-) \le (x_i - a_{ij}) \le 0 \\ 1 - (x_i - a_{ij}) / b_{ij}^+ & \text{if } 0 \le (x_i - a_{ij}) \le (b_{ij}^+) \\ 0, \text{otherwise}; & i \in [1,2];\ j \in [1,...,N_i] \end{cases} \tag{5}$$

where $(x_1 = k_1 e; x_2 = k_2 de/dt)$ or $(x_1 = k_1 |e|; x_2 = k_2 |de/dt|)$; k_1 and k_2 are the positive normalizing factors; $a_{ij}, b_{ij}^-, b_{ij}^+$, and N_i are the centroid, left half-width, right half-width of j^{th} MF, and MF number of input i^{th}, respectively.

By using the layer 3, fuzzy rules based on 'AND' logic, layer 4, defuzzification with singleton MFs, and Remark 1, the crisp output from the layer 4 can be computed as:

$$U_{FA} = \sum_{m=1}^{M} mf(w_m) w_m \Big/ \sum_{m=1}^{M} mf(w_m), (A \text{ is } P, I \text{ or } D) \tag{6}$$

where w_m and M are in turn the weight of MF m^{th} and MF number of the fuzzy A output (then, $1 \le M \le N_1 N_2$); and $mf(w_m)$ is the fuzzy output function given by

$$mf(w_m) = \sum_{j,k} mf_{jk}(w_m) \tag{7}$$

where $mf_{jk}(w_m)$ is defined as the consequent fuzzy output function when the first and second fuzzy inputs are in classes j^{th} and k^{th}.

$$mf_{jk}(w_m) = \delta_{jk} \mu_{jk} \tag{8}$$

where δ_{jk} is an activation factor, which is activated when input x_1 is in class j^{th}, and input x_2 is in class k^{th}; μ_{jk} is height of the consequent fuzzy function obtained from the inputs:

$$\mu_{jk} = \min\left[f_j(x_1), f_k(x_2) \right] \tag{9}$$

Finally, to ensure the boundary condition in (4), the fuzzy tuner output, U_A^{Fuzzy}, is computed using the sigmoid activation function f_{FA} in the layer 5 as

$$U_A^{Fuzzy} \equiv U_A = f_{FA}(U_{FA}) = \left(1 + e^{-U_{FA}}\right)^{-1} \in (0,1), (A \text{ is } P, I \text{ or } D) \tag{10}$$

b) Robust updating rule

Remark 2. In robust control design for a nonlinear system, a family of uncertainties of the system transfer function, *P(s),* needs to be firstly derived, subsequently, obtaining a set of the open-loop

transfer functions as well as a set of the closed-loop transfer functions. In order to ensure a robust control for this system, there are two control objectives. The first is closed-loop robust stability which must be checked with reasonable margins. The robust stability is presented by a forbidden region about the origin which is enclosed by an *M-locus* in the Nichols chart. By the Nyquist criterion, the closed-loop stability is retained as long as the loop gain of the Niquist plot does not cross the critical point q under the uncertainty (the (-1,0) point in the complex plane or the (-180°, 0 dB) point in a Nichols chart). The second control objective is closed-loop disturbance attenuation. For the disturbance rejection requirement, the sensitivity reduction problem must be solved. With no feedback, there is no disturbance modification. Only a high gain feedback loop leads to small sensitivity and to disturbance reduction. Therefore, the upper tolerance is imposed on the sensitivity function. In conditionally stable systems, the complementary sensitivity condition enforces a large loop gain when the plot crosses the 180° line above 0 dB [10].

Transfer function of the PID controller shown in (2) is expressed as

$$G_{PID} = K_P + \frac{K_I}{s} + K_D s \tag{11}$$

The open-loop transfer function set of the system is defined as

$$L(s) = P(s) G_{PID}(s) \tag{12}$$

where $P(s)$ is the family of uncertainties of the system transfer function. This plant transfer function set can be defined by using an input-output data observation of the system open-loop tests and a simple identification toolbox in MATLAB [10].

The sensitivity function determining the set of transfers of the equivalent output disturbance to the controlled output Y can be derived as

$$S(s) = \frac{1}{1 + P(s) G_{PID}(s)} \tag{13}$$

Based on Remark 2, the criteria to select the PID controller gains can be derived. For the robust stability, an approximately minimal value of $M = 1.4$ (3 dB) gain margin for the closed-loop system set is given by

$$\left|\frac{L(s)}{1+L(s)}\right| \le M = 1.4 \tag{14}$$

For the disturbance rejection requirement, the general upper bound of the sensitivity is set to limit the peak value of disturbance amplification as follows:

$$\left|\frac{1}{1+L(s)}\right|_{\max} \le M_D, M_D > 1 \tag{15}$$

Therefore, to ensure that the controller can drive the system to satisfy the performance robustness specifications, the PID gains are updated for each working step of time $(t+1)^{th}$ using the RUR which is defined as follows:

$$K_A(t+1) = \begin{cases} K_A^{Fuzzy}(t+1) \text{ calculated by } (4) \\ \quad \text{IF: } K_A^{Fuzzy}(t+1) \text{ makes } \big((14)\,\&\,(15)\big) \text{ to be satisfied;} \\ K_A(t) \\ \quad \text{IF: } K_A^{Fuzzy}(t+1) \text{ makes } \big((14)\,\&\,(15)\big) \text{ to be not satisfied} \end{cases} \tag{16}$$

From (16), at a working step, the PID controller parameters are then updated as the set given from the fuzzy tuners only if it can ensure the robust stability (14) and disturbance rejection criterion (15). On the other hand, the PID gains are remained the same values as those of the previous working step. Consequently, the main control signal using the designed fuzzy PID-based controller can be robustly computed.

3.2. Robust learning mechanism

Based on the outputs from the RPIGM prediction module, the fuzzy tuners are optimized in advance using the robust learning mechanism to minimize effectively the control error (Fig. 2).

The steepest descent back-propagation (BP) training algorithm using delta rule is widely recognized as one of the most simple but powerful training tools for not only neural network designs but also other applications, for example, fuzzy scheduling systems. However, since it applies the delta rule to update system weighting factors, it suffers from a slow convergence rate and often yield sub-optimal solutions. A variety of approaches have been then applied in an attempt to accelerating the learning process. For example, the standard BP-based system can be improved

by implementing other techniques, such as least square methods, genetic algorithms and particle swam optimisation and Bayesian regularization scheme [36, 37]. Nevertheless, a more intelligent training mechanism normally requires extra computational cost, especially for applications to large and complex systems due to the large number of weighting factors (i.e. for tuning fuzzy inferences with many membership functions). To address these design challenges, this study presents the simple but efficient robust learning mechanism as stated in Remark 3.

Remark 3. The RLM is designed as the combination of a decisive vector size minimizer and the BP algorithm with learning rate adaptation based on Lyapunov stability condition. Here for each fuzzy tuner, the decisive vector size minimizer is firstly used to find out only active input MFs and corresponding active output MFs, which are activated by the fuzzy input values. Next, the BP algorithm using adaptive learning rates is used to optimize these active MFs to minimize an error function defined as

$$E(t+pT)=\frac{1}{2}\left[\hat{y}(t+pT)-y_r(t+pT)\right]^2 \tag{17}$$

where $y_r(t+pT)$ and $\hat{y}(t+pT)$ are in turn the desired and p-step ahead estimated system outputs; T is the sampling period. Without loss of generality, pT can be simplified as p by considering T is a unit of time.

a) Learning algorithm

Based on Remark 3 with the BP algorithm ([9], [17], [19]-[24]) and the designed fuzzy structure (Fig. 2b), the decisive factors of the input and output MFs, $a_{ij}, b_{ij}^-, b_{ij}^+$ and w_m, can be automatically updated for step time $(t+1)^{th}$ using the delta rule as follows:

$$\begin{cases} a_{ij(t+1)} = a_{ijt} - \Delta a_{ijt} = a_{ijt} - \eta_{at}\partial E_{t+p} / \partial a_{ijt} \\ b_{ij(t+1)}^{+/-} = b_{ijt}^{+/-} - \Delta b_{ijt}^{+/-} = b_{ijt}^{+/-} - \eta_{bt}\partial E_{t+p} / \partial b_{ijt}^{+/-} \\ w_{m(t+1)} = w_{mt} - \Delta w_{mt} = w_{mt} - \eta_{at}\partial E_{t+p} / \partial w_{mt} \end{cases} \tag{18}$$

where η_{at}, η_{bt} and η_{wt} are the learning rates within range [0, 1].

- The factor $\partial E_{t+p} / \partial w_{mt}$ in (18) can be calculated using the chain rule:

$$\frac{\partial E_{t+p}}{\partial w_{mt}} = \frac{\partial E_{t+p}}{\partial \hat{y}_{t+p}} \frac{\partial \hat{y}_{t+p}}{\partial u_{PIDt}} \frac{\partial u_{PIDt}}{\partial U_{At}} \frac{\partial U_{At}}{\partial U_{FAt}} \frac{\partial U_{FAt}}{\partial w_{mt}} \tag{19}$$

where, by employing the future system response, $\hat{y}_{t+p}$, estimated by the RPIGM and (17), one has:

$$\frac{\partial E_{t+p}}{\partial \hat{y}_{t+p}} = \hat{e}(t+p) = \hat{y}_{t+p} - y_{r(t+p)}, \tag{20}$$

By simplifying the ratio $\partial \hat{y}_{t+p} / \partial y_t$ as unit, the second term in (19) can be approximated as (21) while the third term in (19) can be computed as (22) using the discrete forms of (2):

$$\frac{\partial \hat{y}_{t+p}}{\partial u_{PIDt}} = \frac{\partial \hat{y}_{t+p}}{\partial y_t} \frac{\partial y_t}{\partial u_{PIDt}} \approx \frac{\Delta \hat{y}_{t+p}}{\Delta u_{PIDt}} = \frac{\hat{y}_{t+p} - \hat{y}_{t+p-1}}{u_{PIDt} - u_{PID(t-1)}} \tag{21}$$

$$\frac{\partial u_{PIDt}}{\partial U_{At}} : \begin{cases} \partial u_{PIDt} / \partial U_{Pt} = \Delta K_P e(t) \\ \partial u_{PIDt} / \partial U_{It} \approx \Delta K_I \sum_{i=1}^{t} e(i) \\ \partial u_{PIDt} / \partial U_{Dt} \approx \Delta K_D \left(e(t) - e(t-1) \right) \end{cases} \tag{22}$$

From (10) and (6), the last two terms in (19) can be derived using the power rule:

$$\frac{\partial U_{At}}{\partial U_{FAt}} = U_{At}(1 - U_{At}) \tag{23}$$

$$\frac{\partial U_{FAt}}{\partial w_{mt}} = \frac{mf(w_{mt})}{\sum_{l=1}^{M} mf(w_{lt})} \tag{24}$$

- Similarly, the factor $\partial E_{t+p} / \partial a_{ijt}$ in (18) can be computed using the chain rule:

$$\frac{\partial E_{t+p}}{\partial a_{ijt}} = \frac{\partial E_{t+p}}{\partial U_{FAt}} \frac{\partial U_{FAt}}{\partial mf(w_{mt})} \frac{\partial f_{ij}(x_t)}{\partial a_{ijt}} \tag{25}$$

where

$$\frac{\partial U_{FAt}}{\partial mf\left(w_{mt}\right)}=\frac{\sum_{l=1}^{M} mf\left(w_{lt}\right)\left(w_{mt}-w_{lt}\right)}{\left(\sum_{l=1}^{M} mf\left(w_{lt}\right)\right)^{2}} \tag{26}$$

$$\frac{\partial f_{ij}\left(x_t\right)}{\partial a_{ijt}}=\begin{cases}-1/b_{ij}^{-} & \text{if } \left(-b_{ij}^{-}\right)\le\left(x_{it}-a_{ij}\right)\le 0\\ 1/b_{ij}^{+} & \text{if } 0\le\left(x_{it}-a_{ij}\right)\le\left(b_{ij}^{+}\right)\\ 0 & \text{otherwise}\end{cases} \tag{27}$$

- And, the factor $\partial E_{t+p}/\partial b_{ij}^{+/-}$ in (18) can be computed by

$$\frac{\partial E_{t+p}}{\partial b_{ij}^{+/-}}=\frac{\partial E_{t+p}}{\partial U_{FAt}}\frac{\partial U_{FAt}}{\partial mf\left(w_{mt}\right)}\frac{\partial f_{ij}\left(x_{it}\right)}{\partial b_{ij}^{+/-}} \tag{28}$$

where

$$\frac{\partial f_{ij}\left(x_t\right)}{\partial b_{ij}^{+/-}}=\begin{cases}-(x_{it}-a_{ij})/(b_{ij}^{-})^{2} & \text{if } (-b_{ij}^{-})\le(x_{it}-a_{ij})\le 0\\ (x_{it}-a_{ij})/(b_{ij}^{+})^{2} & \text{if } 0\le(x_{it}-a_{ij})\le b_{ij}^{+}\\ 0 & \text{otherwise.}\end{cases} \tag{29}$$

b) Lyapunov stability condition

Theorem 1. By selecting properly the learning rates $\eta_{at}=\eta_{bt}$ and η_{wt} for step $(t+1)^{th}$ to satisfy condition (30), then the closed-loop stability of the RPTC control system in Fig. 1 is guaranteed.

$$2e_t\left(\eta_{at}F_2+\eta_{wt}\sum_{l=1}^{M}F_1\frac{\partial E_{t+p}}{\partial w_{lt}}\right)+\left(\eta_{at}F_2+\eta_{wt}\sum_{l=1}^{M}F_1\frac{\partial E_{t+p}}{\partial w_{lt}}\right)^{2}\le 0 \tag{30}$$

where

$$F_1=\frac{mf\left(w_{lt}\right)}{\sum_{m=1}^{M}mf\left(w_{mt}\right)}\Big/\left(\frac{\partial U_{FAt}}{\partial e_t}\right) \tag{31}$$

$$F_2 = \sum_{i=1}^{2}\sum_{j=1}^{N}\left(\pm k_i \frac{\partial E_{t+p}}{\partial a_{ijt}} \pm \frac{\left(x_{it} - a_{ijt}\right)}{k_i b_{ijt}^{+/-}} \frac{\partial E_{t+p}}{\partial b_{ijt}^{+/-}} \right) \tag{32}$$

Proof. See Appendix A. □

Remark 4. There are many solutions to select the learning rates such that condition (30) holds. These rates are initially set to properly small values and then, are tuned using the iterative algorithm based on Theorem 1. The learning rates are kept as constant values when the system is stabilized. Moreover to simply select the learning rates and to provide a continuous transition between input partitions (no dead zone), the MFs of the fuzzy inputs must satisfy following requirements:

- j^{th} MF partition is overlapped by $(j\text{-}1)^{th}$ MF partition
- j^{th} MF partition overlaps $(j+1)^{th}$ MF partition
- $(j\text{-}1)^{th}$ MF partition does not overlap $(j+1)^{th}$ MF partition

Based on Remark 4 and the bounds of fuzzy inputs/outputs (Section 3.1), additional constraints to select the learning rates are derived as

$$\begin{cases} -1 \le a_{i(j-1)} < a_{ij} < a_{i(j+1)} \le 1, \text{or:} 0 \le a_{i(j-1)} < a_{ij} < a_{i(j+1)} \le 1 \\ \left(a_{i(j-2)} + b_{i(j-2)}^{+}\right) \le \left(a_{ij} - b_{ij}^{-}\right) \le \left(a_{i(j-1)} + b_{i(j-1)}^{+}\right), j \ge 3 \\ \left(a_{i(j+1)} - b_{i(j+1)}^{-}\right) \le \left(a_{ij} + b_{ij}^{+}\right) \le \left(a_{i(j+2)} - b_{i(j+2)}^{-}\right), j \le N-2 \\ -10 \le w_m \le 10 \end{cases} \tag{33}$$

The suitable learning rates $\eta_{at} \equiv \eta_{bt}$, and η_{wt} of each fuzzy tuner at step $(t+1)^{th}$ are defined based on Theorem 1 and (33).

c) Decisive vector size minimizer

The decisive parameters of fuzzy PID mechanism are optimized online using the Lyapunov-based learning algorithm to guarantee the control accuracy. However, for each the fuzzy tuner with the two inputs and single output, the more MFs and rules are, the larger the number of decisive factors, $a_{ij}, b_{ij}^{-}, b_{ij}^{+}$ and w_m, is. As the design presented in Section 3.1b, the total number of decisive factors of the three tuners is $9(N_1+N_2)+3M$. For instant, each fuzzy input with five MFs commonly needs twenty five rules for the output and could lead to 165 decisive parameters. The time to train

the controller is therefore considerably increased and subsequently, restricts the applicability of this training method. In order to solve this problem, the decisive vector size minimizer is designed and implemented before the Lyapunov-based learning algorithm to minimize the number of calculations when training the control parameters.

Remark 5. With the fuzzy tuner based on Remark 1 and Remark 4, for a set of values of the inputs (x_1, x_2), it always exits one to maximum two MFs of each input which contain these values (at the left or right side of the MFs). These MFs are called active input MFs (AIMFs). Consequently, the output MFs corresponding to the AIMFs defined by the fuzzy rules are called active output MFs (AOMFs). From Remark 2, for each fuzzy tuner at each working step, only input/output MFs activated by the fuzzy input values, AIMFs and AOMFs, are tuned with respect to the minimization of control error function (17).

From Remark 5, for each step of time with (x_1, x_2) value set, if existing two AIMFs of each fuzzy input, at least one to maximum four AOMFs, corresponding to one to four rules, are selected by the tuner to derive the instantaneous output U_A^{Fuzzy}. In order words, the number of AIMFs and AOMFs can be then listed into four cases in Table 1.

Table 1

Active input MFs and corresponding active output MFs in fuzzy P/I/ Or D tuner.

Number of AOMFs $n_{A\_AOMF;}$ *(A is P/I/ or D)*		Input x_2: Number of AIMFs n_{A_AIMF2}	
		1	2
Input x_1: Number of AIMFs n_{A_AIMF1}	1	1	2
	2	2	4

By utilizing Remark 5, each AIMF has only two parameters, a_j, b_j^- (or b_j^+), and each AOMF has one parameter, w_k, which need to be optimized. Thus for each step of time, the decisive vector size minimizer identifies three dynamic characteristic vectors, denoted as V_{Pt}, V_{It}, and V_{Dt}, for which are then sent to the Lyapunov-based learning algorithm to update the corresponding parameters of the tuners, *P*, *I*, and *D*. From Table 1, sizes of these vectors are determined:

$$\begin{cases} \left(V_{At}\right)_{\max size} = \text{Size of } \left[a_{11}\ b_{11}^{-/+}\ a_{21}\ b_{21}^{-/+}\ a_{12}\ b_{12}^{-/+}\ a_{22}\ b_{22}^{-/+}\ w_1\ w_2\ w_3\ w_4\right]_t = \left[1\times 12\right] \\ \left(V_{At}\right)_{\min size} = \text{Size of } \left[a_{11}\quad b_{11}^{-/+}\quad a_{21}\quad b_{21}^{-/+}\quad w_1\right]_t = \left[1\times 5\right], A: P, I \text{ or } D. \end{cases} \tag{34}$$

From (34), the total number of parameters which needs to be tune each step is reduced from $(9(N_1+N_2)+3M)$ to the range [15, 36] disregarding the number of input/output MFs. Thus, the decisive vector size minimizer could save remarkably the time to optimize the control parameters.

4. Robust PI grey model-based prediction module

The RPIGM prediction module comprises two RPIGM predictors in which each predictor is constructed based on a main grey model (MGM, Section 4.1) integrated a PI-based weight tuner (PIWT, Section 4.2) to perform the prediction. Here, the PIWT is designed based on a Lyapunov stability condition and employed to regulate online the weight factors of the MGM with respect to the prediction error minimisation in order to ensure a robust prediction. The prediction procedure using the proposed RPIGM is then demonstrated in Fig. 3.

4.1. Main grey model

In grey prediction theory, GM(n, m) denotes the grey model, where n and m are the order and number of variables of the model. And GM(1,1) is known as the most popular grey model used for many practical applications [26]-[35]. With only a few historical data of the system output(s), the grey predictor can predict the future output(s) without knowing the mathematical model of the real system.

To estimate a system behaviour, at first, a typical grey predictor conducts an accumulated generating operation (AGO) on an original data sequence of this behaviour. The new series obtained from the AGO is then used to generate a background series based on a mean generating operation (MGO). The resultant series is used to establish a differential equation to calculate optimal model coefficients via the least-square method (LSM). Next using these optimal coefficients, the accumulated generating series of the prediction model are obtained. These values can be returned to estimate the future system behaviour in the time-domain by using an inverse accumulated generating operation (IAGO) and/or recursive operation.

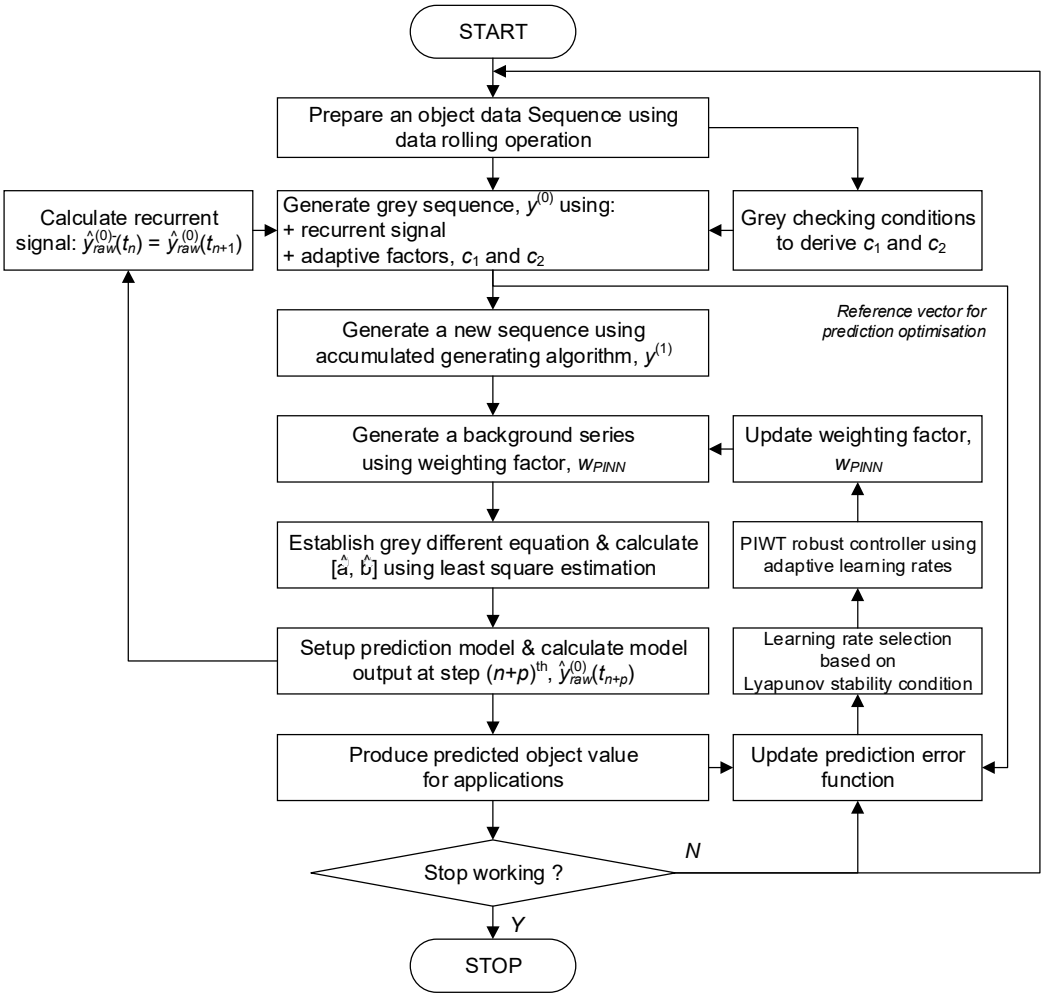


Fig. 3. RPIGM prediction procedure using the MGM(1,1) integrated PIWT.

However, there exist some limitations in typical grey models which affect directly to their applicability as well as prediction accuracy:

- Input raw data must be non-negative and satisfies the smooth condition (will be discussed later) to perform the grey model [25].
- The background series of the model obtained using the MGO could cause internal errors in constructing the grey model [35].
- There is no such condition or constraint to guarantee the model robustness.

Acknowledging these challenges, without loss of generality, the MGM is designed in this study with single variable and first-order type, MGM(1,1), and integrated with the PIWT to estimate

precisely any signal, y, with random distribution. Using the grey theory, the MGM-based prediction procedure is built as demonstrated in Fig. 3 and, can be expressed as the following steps:

Step 1: Prepare an input grey sequence capable of approximating any system:

- Prepare an sequence containing at least five latest data points of the object using the data rolling operation (discarding an oldest data and adding a newest data for each cycling time, [25]):

$$\left\{y_{Object}\left(t_{O1}\right), y_{Object}\left(t_{O2}\right), ..., y_{Object}\left(t_{Om}\right)\right\};\ m \geq 5 \tag{35}$$

- Generate a raw input grey sequence with equal intervals from (35):

$$y_{raw}^{(0)} = \left\{y_{raw}^{(0)}\left(t_1\right), y_{raw}^{(0)}\left(t_2\right), ..., y_{raw}^{(0)}\left(t_n\right)\right\};\ n \geq 5 \tag{36}$$

here, if the condition (37) holds, then sequence (36) is exactly similar to sequence (35); otherwise, sequence (36) is the sequence containing sequence (35) and new elements up to latest time point with the same intervals that are derived as (38) using the same MGM-based prediction.

$$\begin{aligned} &\Delta t_{Oi} / T_{RGM} < 2;\ \Delta t_{Oi} = t_{Oi} - t_{O(i-1)};\ \forall i \in [2, ..., m] \\ &\Delta t_{O1} = t_{O1} - t_{Olast}; t_{Olast} : \text{time of previous value of } y_{Object}\left(t_{O1}\right); \end{aligned} \tag{37}$$

here T_{MGM} is the desired prediction sampling period;

$$\begin{aligned} &y_{raw}^{(0)}\left(t_k\right) = \hat{y}_{raw}^{(0)}\left(t_k\right);\ t_{Oi} \leq t_k \leq t_{O(i+1)}; \\ &k = 2, ..., n;\ n = \left\lfloor \left(t_{Om} - t_1\right) / T_{RGM} \right\rfloor; i = 1, ..., m-1; \end{aligned} \tag{38}$$

here $\lfloor * \rfloor$ is the floor function to return nearest integer value.

- To exhibit the dynamic temporal behaviour of the model, a recurrent signal as the one-step-ahead predicted value, denoted as $\hat{y}_{raw}^{(0)-}\left(t_n\right) \equiv \hat{y}_{raw}^{(0)}\left(t_{n+1}\right)$, is added to the sequence. Thus, (36) becomes:

$$\begin{aligned} &y_{raw}^{(0)} = \left\{y_{raw}^{(0)}\left(t_1\right), y_{raw}^{(0)}\left(t_2\right), ..., y_{raw}^{(0)}\left(t_n\right)\right\};\ n \geq 6 \\ &y_{raw}^{(0)}\left(t_n\right) \equiv \hat{y}_{raw}^{(0)-}\left(t_n\right). \end{aligned} \tag{39}$$

- Remove the oldest elements from sequences (35) and (36). Then, generated a new series (40) from (39) using Remark 6 to make it satisfy grey sequence checking conditions (41).

$$\begin{aligned} &y^{(0)} = \{y^{(0)}(t_1), y^{(0)}(t_2), ..., y^{(0)}(t_n)\};\ n \geq 5 \\ &y^{(0)}(t_k) = y^{(0)}_{raw}(t_k) + c_1 + c_2;\ k = 1, ..., n \end{aligned} \tag{40}$$

$$\begin{cases} y^{(0)}(t_k) > 0 \\ \dfrac{y^{(0)}(t_k)}{\sum_{i=1}^{k-1} y^{(0)}(t_i)\Delta t_i} \leq \dfrac{1}{2},\ \forall k = 1, ..., n \end{cases} \tag{41}$$

Remark 6. By adding two non-negative additive factors c_1 and c_2 derived from (42) using the previous work [35], the sequence (39) becomes (40) and satisfies the grey sequence checking conditions (41).

$$\begin{cases} c_1 = \xi + \max\limits_{k=1,..,n} \left\{ \left| y^{(0)}_{raw}(t_k) \right|, y^{(0)}_{raw}(t_k) \leq 0 \right\} \\ c_2 \geq \max\limits_{k=1,..,n} \left(0, \left\{ \dfrac{2\left(y^{(0)}_{raw}(t_k) + c_1\right)\Delta t_k - \sum_{i=1}^{k-1}\left(y^{(0)}_{raw}(t_i) + c_1\right)\Delta t_i}{\sum_{i=1}^{k-1}\Delta t_i - 2\Delta t_k} \right\} \right) \end{cases} \tag{42}$$

here, ξ is the small positive constant.

Step 2: Generate a new series $y^{(1)}$ from $y^{(0)}$ using the AGO:

$$\begin{aligned} y^{(1)}(k) &= \sum\nolimits_{i=1}^{k} y^{(0)}(i) \times \Delta t_k, k = 1, 2, ..., n \\ &= \sum\nolimits_{i=1}^{k} \left(y_{raw}(i) + c_1 + c_2\right) \times \Delta t_k \end{aligned} \tag{43}$$

Step 3: Instead of employing the MGO in typical grey models, the background series $z^{(1)}$ is newly built from $y^{(1)}$ as:

$$z^{(1)}(t_k) = w_k y^{(1)}(t_k) + w_{k-1} y^{(1)}(t_{k-1}); k = 2, ..., n \tag{44}$$

where $\{w_k, w_{k-1}\}$ are the set of weight factors which are designed as:

$$\begin{cases} w_k &= 0.5\left(1-\delta_1\right)+\delta_1\delta_2 w_{PINN}+\delta_1\left(1-\delta_2\right)\left(1-w_{PINN}\right) \\ w_{k-1} &= 0.5\left(1-\delta_1\right)+\delta_1\left(1-\delta_2\right) w_{PINN}+\delta_1\delta_2\left(1-w_{PINN}\right) \end{cases} \tag{45}$$

where w_{PINN} is called the adaptive gain and $0 < w_{PINN} < 1$; in order to ensure the robust prediction, this adaptive gain is online regulated by the PIWT (Fig. 3) that is introduced in the next section; $\{\delta_1, \delta_2\}$ is the set of activation factors and given as

$$\{\delta_1,\delta_2\} = \begin{bmatrix} \{1,0\}, \text{IF}: y_{raw}^{(0)}\left(t_k\right) \equiv \hat{y}_{raw}^{(0)}\left(t_k\right); y_{raw}^{(0)}\left(t_{k-1}\right) \equiv y_{Object}\left(t_{k-1}\right) \\ \{1,1\}, \text{IF}: y_{raw}^{(0)}\left(t_k\right) \equiv y_{Object}\left(t_k\right); y_{raw}^{(0)}\left(t_{k-1}\right) \equiv \hat{y}_{raw}^{(0)}\left(t_{k-1}\right) \\ \{0,1\}, \text{Others} \end{bmatrix} \tag{46}$$

Step 4: Establish the grey differential equation [25]

$$y^{(0)}\left(t_k\right) + a z^{(1)}\left(t_k\right) = b \tag{47}$$

where *a* and *b* are the model parameters. By employing the LSM [25], the optimal values of the model parameters are obtained:

$$\hat{\beta}_{ab} = \begin{bmatrix} \hat{a} & \hat{b} \end{bmatrix}^T = \left(B^T B\right)^{-1} B^T Y \tag{48}$$

with $B = \begin{bmatrix} -z^{(1)}\left(t_2\right) & 1 \\ -z^{(1)}\left(t_3\right) & 1 \\ \vdots & \vdots \\ -z^{(1)}\left(t_n\right) & 1 \end{bmatrix}$, $Y = \begin{bmatrix} y^{(0)}\left(t_2\right) \\ y^{(0)}\left(t_3\right) \\ \vdots \\ y^{(0)}\left(t_n\right) \end{bmatrix}$.

Step 5: By replacing the optimal solution (48) into (47) and using the recursive operation, the MGM(1,1) prediction is setup as follows:

$$\hat{y}^{(0)}\left(t_k\right) = \frac{\hat{b} - \hat{a} y^{(0)}\left(t_1\right)\Delta t_1}{1+\alpha\left(t_2\right)\hat{a}\Delta t_2} \prod_{i=3}^{k} \frac{1+\left(\alpha\left(t_{i-1}\right)-1\right)\hat{a}\Delta t_{i-1}}{1+\alpha\left(t_i\right)\hat{a}\Delta t_i} \tag{49}$$

Step 6: From (40) and (49), the predicted value of *y* at step $(n+p)^{th}$ can be computed:

$$\hat{y}_{raw}^{(0)}\left(t_{n+p}\right)=\frac{\hat{b}-\hat{a}y^{(0)}\left(t_1\right)\Delta t_1}{1+\alpha\left(t_2\right)\hat{a}\Delta t_2}\prod_{i=3}^{n+p}\frac{1+\left(\alpha\left(t_{i-1}\right)-1\right)\hat{a}\Delta t_{i-1}}{1+\alpha\left(t_i\right)\hat{a}\Delta t_i}-c_1-c_2 \tag{50}$$

where p is the step size of the grey predictor.

4.2. PI-based weight tuner

In this section, the PI-based weight tuner is designed as the combination between PI algorithm, neural network and Lyapunov stability condition and integrated to the MGM to optimize online the adaptive gain w_{PINN} in order to ensure the robust prediction.

Remark 7. By considering the MGM-based prediction as a tracking control problem as described in Fig. 4, it is possible to derive a 'robust controller' PIWT to guarantee the closed-loop prediction stability of the 'plant' MGM.

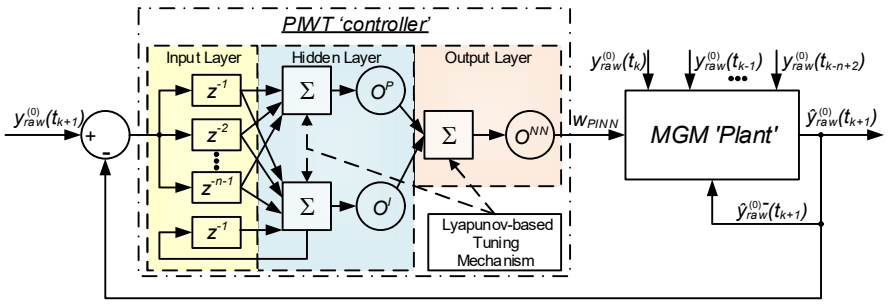


Fig. 4. Representative closed-control form to optimize the MGM using the PIWT.

Based on Remark 7 and Fig. 4, the PIWT is designed as the PI-type neural network structure with the Lyapunov stability condition. Here, the network consists of three layers: an input layer containing the prediction error sequence $\{y_{raw}^{(0)}-\hat{y}_{raw}^{(0)}\}$, a hidden layer with two nodes following PI algorithm, and an output layer to compute the adaptive gain, w_{PINN}. Define $e_k^{MGM}=\{e_1,...,e_{n-1}\}$ $(e_i=y_{raw}^{(0)}(t_{k-i+1})-\hat{y}_{raw}^{(0)}(t_{k-i+1}),i=1,...,n-1)$ is the input vector at step k^{th}, $\{w_k^{Pi},w_k^{Ii}\}$ are the weight vectors of the hidden node P and I, respectively, and, $\{w_k^P,w_k^I\}$ are the weights of the output layer. Since, the output from each hidden node is derived as

$$\begin{cases} O_k^P = \sum_{i=1}^{n-1} w_k^{Pi} e_k^i : \text{Node P} \\ O_k^I = O_{k-1}^I + \sum_{i=1}^{n-1} w_k^{Ii} e_k^i : \text{Node I} \end{cases} \tag{51}$$

Then, the output from the network is obtained using the PI algorithm and sigmoid activation function:

$$w_{PINN} = f\left(O_k^{NN}\right) = \frac{1}{1+e^{-O_k^{NN}}}, O_k^{NN} = w_k^P O_k^P + w_k^I O_k^I \tag{52}$$

The 'controller' output (52) is then fed into the 'plan' MGM (Section 4.1) to perform the prediction (as described in Fig. 3).

Next in order to ensure the robust prediction, the BP training algorithm based on a Lyapunov stability condition is utilized to tune the PIWT weights. Define a prediction error function:

$$E_k^{MGM} = 0.5\sum_{i=1}^{n-1}\left(y_{raw}^{(0)}\left(t_{k-i+1}\right) - \hat{y}_{raw}^{(0)}\left(t_{k-i+1}\right)\right)^2 = 0.5\sum_{i=1}^{n-1}\left(e_k^i\right)^2 \tag{53}$$

Thus, the weight factors of the PIWT are online tuned for the next prediction step, $(k+1)^{th}$, using the delta rule:

$$\begin{cases} w_{k+1}^{P/I} = w_k^{P/I} - \eta_k^{P/I} \partial E_k^{MGM} / \partial w_k^{P/I} \\ w_{k+1}^{Pi/Ii} = w_k^{Pi/Ii} - \eta_k^{Pi/Ii} \partial E_k^{MGM} / \partial w_k^{Pi/Ii} \end{cases} \tag{54}$$

where $\eta_k^{P/I}, \eta_k^{Pi/Ii}$ are learning rates within range [0,1]; the other factors in (54) are derived using the partial derivative of the error function with respect to each decisive parameter.

Theorem 2. By selecting properly the learning rates $\eta_k^{Pi/Ii} = \eta_k^{P/I} = \eta_k$ for step $(k+1)^{th}$ to satisfy (55), the stability of the MGM(1,1) prediction is guaranteed.

$$\sum_{i=1}^{n-1}\left(e_k^i F_k \eta_k + 0.5 F_k^2 \eta_k^2\right) \le 0 \tag{55}$$

with

$$F_k = -\sum_{j=1}^{n-1}\left(\frac{e_k^j}{w_k^{Pi}}\frac{\partial E_k^{MGM}}{\partial w_k^{Pj}} + \frac{e_k^j}{w_k^{Ii}}\frac{\partial E_k^{MGM}}{\partial w_k^{Ij}}\right).$$

Proof. Define a Lyapunov function as

$$V_k^{MGM} = 0.5\sum_{i=1}^{n-1}\left(y_{raw}^{(0)}(t_i) - \hat{y}_{raw}^{(0)}(t_i)\right)^2 = 0.5\sum_{i=1}^{n-1}\left(e_k^i\right)^2 \tag{56}$$

By using the same method as presented in Appendix A according to (56), the MGM(1,1) prediction is guaranteed to be stable only if $\Delta V_{k+1}^{MGM} \le 0$. By selecting properly the learning rate to satisfy (55), the sufficient condition for $\Delta V_{k+1}^{MGM} \le 0$ is guaranteed. Therefore the proof is completed.

4.3. Fuzzy grey cognitive map decision tool for RPIGM

A fuzzy cognitive map (FCM) is known as the neuro-fuzzy system which is graphically represented by a frame of nodes (input and output concepts) and connection edges between nodes to be capable of incorporating knowledge from experts [38]-[42]. With a traditional FCM, intensity of a causal relation between two concepts (weight) is set to zero in the adjacency matrix if this relation does not exist or is partial/completely unknown. Thus, a combination between FCMs and grey numbers can perform an effective decision making tool for solving problems within environments with high uncertainties and/or incomplete information.

In this study, an adaptive fuzzy grey cognitive map is newly designed in which the grey weights with dynamic bounds are used to build the map and online adjusted to minimize a pre-defined cost function. This FGCM is then used to tune the RPIGM prediction step size.

a) FGCM design

By using the grey theory [25], the FGCM is generally designed for a set of N^C concepts and, therefore, can be represented by

$$\left\{\otimes C_i(t), \otimes w_{ij}(t), f^C\right\} \tag{57}$$

where $\otimes C_i(t)$ is the grey value of i^{th} concept defined in (58); $\otimes w_{ij}(t)$ is the grey weight between concept i^{th} and j^{th} defined in (59); $f^C(*)$ is the activation function defined in (60) (λ is the steepness parameter).

$$\otimes C_i(t) \in [\underline{C}_i(t), \overline{C}_i(t)] \in ([0,1] \text{or} [-1,1]), \tag{58}$$

$$\otimes w_{ij}(t) \in [\underline{w}_{ij}(t), \overline{w}_{ij}(t)] \in ([0,1] \text{ or } [-1,1]), \tag{59}$$

$$f^C(*) = \begin{cases} (1+e^{-\lambda(*)})^{-1} & , if\ (*) \in [0,1], \\ (e^{\lambda(*)} - e^{-\lambda(*)})(e^{\lambda(*)} + e^{-\lambda(*)})^{-1}, & if\ (*) \in [-1,1]. \end{cases} \tag{60}$$

Remark 8. Bounds of grey numbers in the FGCM (concepts' values and weights) are properly initialized within the maximum range [0,1] or [-1,1]. These bounds are adjustable online but limited to their maximum ranges.

By using FCM theory and Remark 8, the grey value of each concept can be updated for each step of time based on the influences of the other interconnected concepts:

$$\begin{aligned} \otimes C_i(t+1) &= f^C\left(\otimes C_i(t) + \sum\nolimits_{j=1, j\neq i}^{N} \otimes w_{ji}(t) \otimes C_j(t)\right) \\ &= sat\left(\underline{C}_i(t+1), \overline{C}_i(t+1)\right) \in \left([0,1] \text{ or } [-1,1]\right) \end{aligned} \tag{61}$$

where *sat*(*) is the saturation function of *.

b) FGCM training algorithm

In order to increase the decision accuracy of the FGCM when dealing with partial unknown systems and uncertainties, the grey weights are updated online using the nonlinear Hebbian-type learning rule [42]:

$$\begin{aligned} \otimes w_{ji}(t+1) &= \otimes w_{ji}(t) + \Delta \otimes w_{ji}(t) \\ &= \otimes w_{ji}(t) + \eta_{AFGCM} \otimes C_j(t)\left(\otimes C_i(t) - \otimes w_{ji}(t) \otimes C_j(t)\right) \\ &= sat\left(\underline{w}_{ji}(t+1), \overline{w}_{ji}(t+1)\right) \in \left([0,1] \text{ or } [-1,1]\right) \end{aligned} \tag{62}$$

here η_{AFGCM} is the learning rate.

Remark 9. The FGCM grey weights are updated until one of the two following termination conditions are achieved:

- Cost function minimization condition:

$$J = \sum_{i=1}^{m} \left(C_{d_i}^{des} - C_i^{des} \right)^2 \le J_{\min},\ J_{\min} > 0 \tag{63}$$

- Minimum changing speed of decisive concepts' values:

$$\left| C_i^{des}(t+1) - C_i^{des}(t) \right| \le \delta_C, \forall i = 1,...,m,\ \delta_C > 0 \tag{64}$$

where $C_{d_i}^{des}$ is the desired value of the decisive concept i^{th}, C_i^{des} ; m is the number of decisive concepts; J_{min} is the desired cost; δ_C is the small constant.

4.4. Utilization of RPIGM with FGCM for system response estimation

By utilizing the designs of MGM(1,1), PIWT and FGCM, the first predictor, tagged as RPIGM1, is constructed to derive the filtered value of the system response at the current step, $\hat{y}(t) \equiv \hat{y}_{raw}^{(0)}(t_n)$, and its estimated value at p-step ahead, $\hat{y}(t+p) \equiv \hat{y}_{raw}^{(0)}(t_{n+p})$. The filtered response is sent to the second predictor, tagged as RPIGM2, to produce the compensative control signal while the predicted future response is sent to the RFPID module to optimize its control parameters (Section 3). Generally, the purpose is to design a controller to achieve a good tracking performance. In other words, the control system should be built in order to speed up the system response (or reduce the rising time), and reduce the steady state error (or increase the control precision).

Remark 10. In a control system using grey predictor, the prediction step size (p) affects directly on the system performance. During the rising time period, with a small value of p the predictor speeds up the system response but causes the large overshoot or oscillation. Otherwise when the system is closed to the desired state, the predictor with a large value of p reduces the overshoot but increases the rising time [16].

Based on Remark 10, the FGCM with three concepts is applied to the PRIGM1 to tune online the prediction step size, p. In this case, the tracking control error (e) and its derivative $(\dot{e})$ are the input concepts while the output concept is p. Please note that according to the tracking error, the step size p is continuously regulated using the principle in Remark 10 to improve the control performance. The grey weight training is only terminated once the weights reach to a stable pattern.

4.5. Utilization of RPIGM for control compensation of noises and disturbances

As presented in Section 4.3, the RPIGM2 receives the filtered response, output from the RPIGM1, to estimate the influences of noises and disturbances on the system performance at the following step and, consequently, to create properly the additive correction to the main control signal (as in (1)) to compensate for these undesirable influences.

Here, the data sequence input to the model is the amount of system response, e_{ND}, caused by the noises and disturbances as

$$\begin{cases} e^{(0)}_{ND\_raw} = \left\{ e^{(0)}_{ND\_raw}\left(t_1\right), e^{(0)}_{ND\_raw}\left(t_2\right), ..., e^{(0)}_{ND\_raw}\left(t_{n_{ND}}\right) \right\},\ n_{ND} \geq 5 \\ e^{(0)}_{ND\_raw}\left(t_k\right) = \hat{y}^{(0)}_{raw}\left(t_k\right) - y^{(0)}_{raw}\left(t_k\right),\ k = 1, 2, ..., n_{ND} \end{cases} \tag{65}$$

here $y^{(0)}_{raw}(t_k)$ and $\hat{y}^{(0)}_{raw}(t_k)$ are in turn the measured system response $y(t)$ and its filtered value $\hat{y}(t)$ using the RPIGM1 at step k^{th}.

The impacts of noises and disturbances on the system at the coming step of time, $(n+1)^{th}$, is then estimated using (50) as

$$\hat{e}^{(0)}_{ND\_raw}\left(t_{n_{ND}+1}\right) = \frac{\hat{b}_{ND} - \hat{a}_{ND} e^{(0)}_{ND}\left(t_1\right)\Delta t_1}{1 + \alpha_{ND}\left(t_2\right)\hat{a}_{ND}\Delta t_2} \prod_{i=3}^{n_{ND}+1} \frac{1 + \left(\alpha_{ND}\left(t_{i-1}\right) - 1\right)\hat{a}_{ND}\Delta t_{i-1}}{1 + \alpha_{ND}\left(t_i\right)\hat{a}_{ND}\Delta t_i} - c_{ND1} - c_{ND2} \tag{66}$$

Finally, the additive control signal with respect to the estimated influence of perturbations on the system at the coming step, $(t+1)^{th}$, shown in (3) can be derived by using the result in (66).

5. Illustrative Examples

5.1. RPIGM prediction module evaluation

a) Case study 1 – Prediction accuracy

At first, the capability of the developed RPIGM predictor in estimating an arbitrary signal has been investigated. Here, a comparative study of three grey models, a typical GM(1,1) (shortened as GM) [16], the SAUIGM(1,1) (shortened as SAUIGM) [35] and the RPIGM (the PIWT-

integrated MGM(1,1)), was carried out to perform one-step ahead estimation (p = 1) of communication delay problem in the networked system introduced in [35]. The functionalities of these models can be summarized in Table 2.

Table 2

Functionalities of comparative grey models.

Grey model	Functionalities			
	Input data	Data intervals	Background series	Prediction error minimisation
GM	Only data satisfying checking conditions	Must be equal intervals	Use MGO	No
SAUIGM	Work with any data	Work with unequal intervals	Use modified MGO (exact solution)	Use *l*th-order error correction accumulation
RPIGM	Work with any data	Work with unequal intervals	Use modified algorithm with adaptive weights	Use PIWT with Lyapunov stability condition

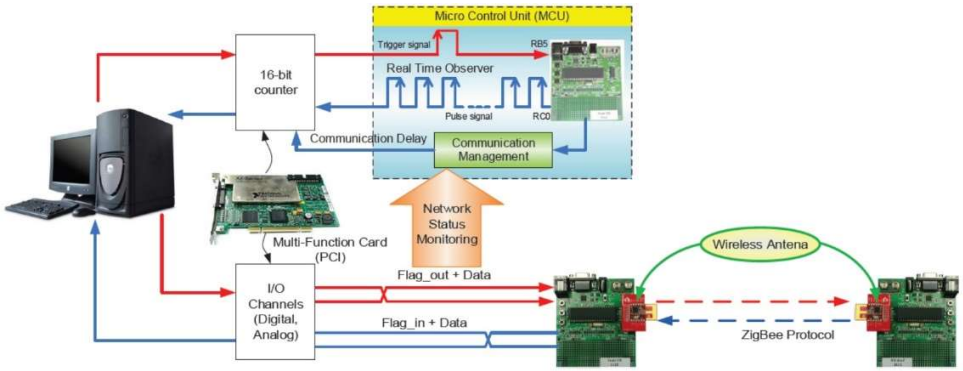


Fig. 5. Experimental setup to detect communication delays of the networked system [35].

The experimental setup used to investigate the networked system delays is demonstrated in Fig. 5. As shown in this figure, the networked system consists of one coordinator and one router based on ZigBee protocol, and the wireless series from Microchip Technology Inc.—MRF24J40MA with the operating frequency of 2.4 GHz was employed [35]. The coordinator also can exchange data with a personal computer (PC) through an NI multi-function card, PCI-6251. In addition, a time delay measurement (TDM) module employing a micro control unit (MCU) was installed and connected to the PC through the multi-function board. Here, the MCU with timer and counter

functions is capable of, first, measuring accurately the real working time of the system and, second, recognizing the networked communication delays [35].

To investigate only the communication delay through this networked setup and due to the applicability of the GM limited to equal-time-interval time series, a simple program with the fixed sampling period of 10ms was built in the Simulink environment combined with Real-time Windows Target Toolbox of Matlab to send a random signal to the coordinator during 30 seconds. This signal was then transmitted to the router by the ZigBee protocol. Once receiving the information from the coordinator, the router immediately sent a random signal back to the coordinator to perform the simple closed-loop networked system. The three grey models were then applied to forecast the time delays over the testing period.

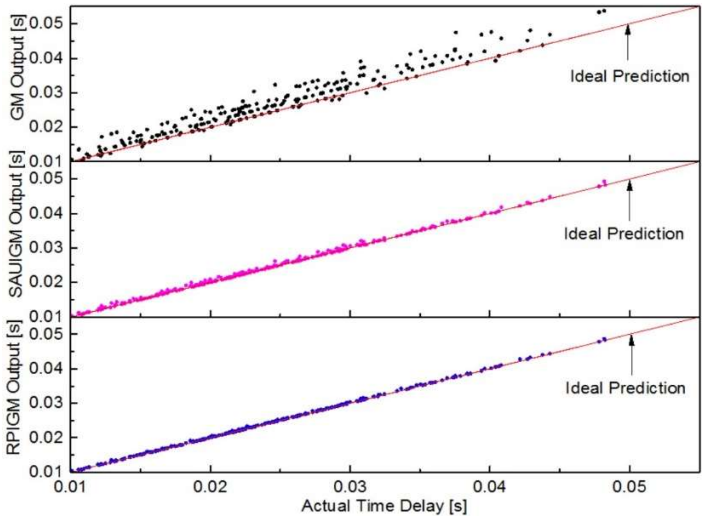


Fig. 6. Real-time delay predictions vs. actual delays.

Table 3

Real-time delay predictions using different models.

Evaluation criteria	Prediction models		
	GM	SAUIGM	RPIGP
ARE (%)	8.6720	0.9664	0.4783
RMSE (10^{-4})	28.4937	3.7635	1.8398
R2	0.8818	0.9679	0.9891

The one-step ahead prediction results were then obtained and presented by the scatter plots of the predicted delays versus the actual delays in Fig. 6. The ideal prediction, with 100% prediction accuracy, was represented by the red linear lines. From the results, it is clear that the prediction

accuracy using the typical GM as shown in the top sup-plot of Fig. 6 was low due to its drawbacks (stated in Section 4.1 and Table 2). By applying the SAUIGM with the exact solution to compute the background series and *l*th-order error correction accumulation [35], the performance was significantly improved (see the middle sub-plot). However, there was no sufficient condition to ensure the robust prediction. Meanwhile, by employing the RPIGM, which was constructed based on the PIWT using the PI-based neural network and Lyapunov condition, the robust prediction with the highest accuracy could be achieved as presented in the bottom sub-plot.

Three evaluation criteria, average relative error (*ARE*), root mean square error (*RMSE*), coefficient of determination (*R2*), were employed to evaluate the performances of the grey models:

$$ARE(\%) = \frac{1}{n}\sum_{k=1}^{n}\left(\frac{\left|y_{raw}^{(0)}(k) - \hat{y}_{raw}^{(0)}(k)\right|}{y_{raw}^{(0)}(k)} \times 100\right) \tag{67}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{k=1}^{n}\left(y_{raw}^{(0)}(k) - \hat{y}_{raw}^{(0)}(k)\right)^2} \tag{68}$$

$$R2 = 1 - \frac{\sum_{k=1}^{n}\left(y_{raw}^{(0)}(k) - \hat{y}_{raw}^{(0)}(k)\right)^2}{\sum_{k=1}^{n}\left(y_{raw}^{(0)}(k) - \overline{y}\right)^2} \tag{69}$$

where y in this case if the communication delay; $\overline{y}$ is the mean value of this delay observation.

The model evaluation was then carried out as shown in Table 3. The results confirmed that the best prediction performance was enhanced by the RPIGM.

b) Case study 2 – Prediction stability

To assess the stability performances of the three compared models, a series of predictions of the networked communication delay observed in the case study 1 has been performed using these models in which the prediction step sizes were varied from small to large values. After obtaining

the estimation results, a model fitness evaluation versus different values of the prediction step size (p) was done as plotted in

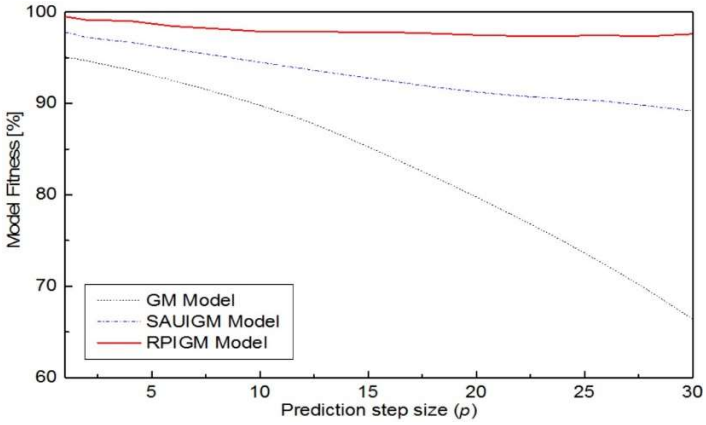


Fig. 7.

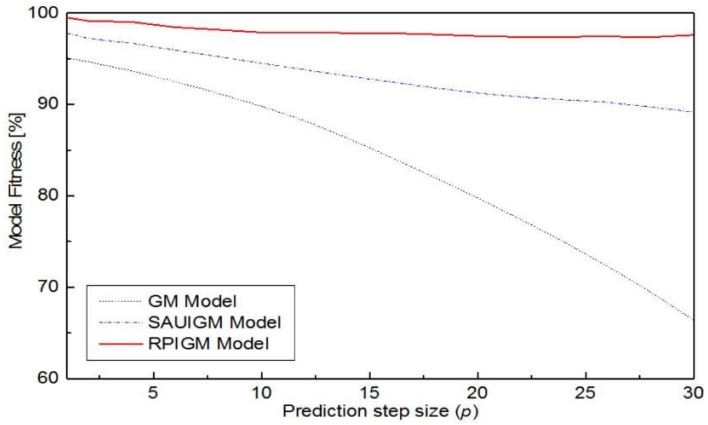


Fig. 7. Fitness evaluation on the delay prediction models.

The result shows that the GM-based estimation accuracy was continuously reduced according to the step size increase (indicated by the black dot line). Although the SAUIGM could improve the accuracy with the error correction accumulation (the blue dash-dot line), its performance was also degraded due to the lack of stability constraint. Only by utilizing the RPIGM with the robust stability condition, the high accuracy could be well maintained as (the red solid line) or, the stable estimation was guaranteed.

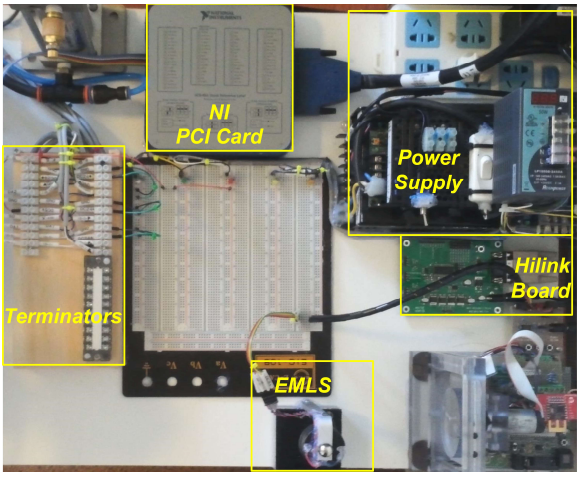
5.2. RPTC tracking control evaluation

In order to validate the ability of the RPTC approach in actual applications, real-time tracking control of an electro-magnetic levitation system (EMLS, [43]) as depicted in Fig. 8 has been investigated. This system is highly nonlinear, open-loop unstable and extremely challenging to

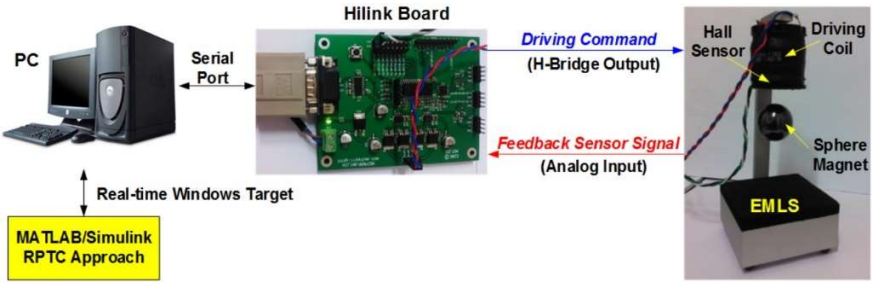
control robustly. The EMLS includes a sphere barium-strontium magnet (2.54mm of diameter), a ferrite-core coil to position the magnet, and a 50V/T Hall effect sensor to observe the magnet position. The driving coil and hall sensor are installed vertically and fixed on a frame to create the moving space for the magnet. The magnet can be levitated in mid-air with an impressive air gap about 25mm using a Hilink real-time multi-function card [44].

The Hilink platform offers a seamless interface between a physical plant and Matlab/Simulink via a serial communication port. This is fully integrated into Matlab/Simulink and comes with a specific Hilink library blocks associated with hardware inputs and outputs. It therefore allows quick configuration of control strategies in real-time with a real plant in the loop. The platform achieves real-time operation with sampling rates up to 3.8kHz [44]. Here, the EMLS control signal is sent from the high performance PC installed Matlab/Simulink through the Hilink board. The magnet movement is managed and represented by the Hall sensor signal which is fed-back to the PC through the card to perform the closed-loop control system.

To evaluate the control effectiveness, the proposed RPTC scheme was compared with other four control methods in driving the magnet of the EMLS. The compared controllers were a PID controller and three advanced controllers: fuzzy PID (FPID), fuzzy PID based on an online tuning grey predictor [16] (OTGFPID1), and online tuning fuzzy PID based on an online tuning grey predictor in which the grey model was the SAUIGM model developed in [35] (OTGFPID2). The advantages of these control methodologies over conventional techniques have been proven in [9] and [16].

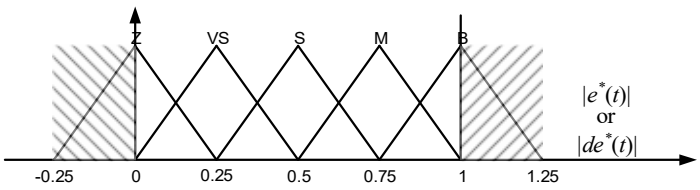


(a) Experiment platform for EMLS tracking control tests

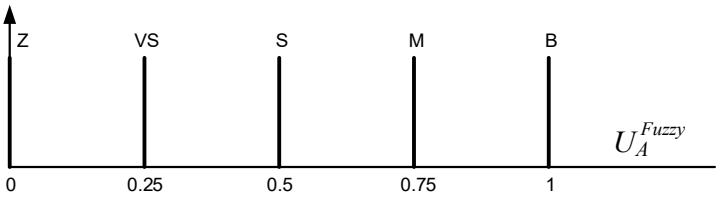


(b) Configuration of the EMLS

Fig. 8. Configuration of EMLS system with real-time tracking control.



(a) Initial MFs for the inputs: $\left|e^*(t)\right|, \left|de^*(t)\right|$



(b) Initial MFs for the output: U_A^{Fuzzy} (A is P, I, or D)

Fig. 9. Initial MFs of the inputs/outputs of fuzzy tuners: P, I, or D.

For simplicity in making the comparison with the developed controllers, the fuzzy tuner designs of the RFPID control module of the RPTC were followed the designs in [9] and [16]. Thus, each the fuzzy tuner contained two inputs, $\left|e^*(t)\right|$ and $\left|de^*(t)\right|$, and one output U_A^{Fuzzy} (A is P, I, or D). For each fuzzy input as well as output, five MFs, tagged "Z", "VS", "S", "M" and "B", were used for smoothly operation while it does not require much calculating time consumption. For the initial state, these input and output MFs were positioned at the same intervals as in Fig. 9a and Fig. 9b, respectively. The rules for the fuzzy tuners were derived as in Table 4. Furthermore to form the RUR, the family of uncertainties of the system transfer function was obtained directly from the modelling results published in [43].

Next, the designs and setting of the four comparative controllers were considered. With the PID controller, the PID gains were derived through a two-step procedure in Matlab/Simulink: first, the EMLS model developed in [43] was employed to represent the real system and their model parameters were optimized using the parameter estimation toolbox, and second, a closed-loop control simulation with the optimized model and the PID controller was performed to optimize the PID gains using the PID tuning toolbox. The last FPID, and OTGFPID1 and OTGFPID2 controllers were constructed with the same fuzzy PID design as that of the RPTC except the use of the robust learning mechanism (Section 3.2). In addition, the fuzzy PID parameters of the OTGFPID2 was online tuned by the delta rule-based learning mechanism in [9]. For the prediction functions, the typical grey model, GM(1,1), of OTGFPID1 and the SAUIGM model of OTGFPID2 used the same method proposed in [16] to tune the prediction step size.

Table 4

Rules table of RFPID control module.

Fuzzy Outputs $(U_P^{Fuzzy}, U_I^{Fuzzy}, U_D^{Fuzzy})$		$\lvert de^*(t)\rvert$				
		Z	VS	S	M	B
$\lvert e^*(t)\rvert$	Z	VS,B,M	VS,B,M	Z,B,M	Z,B,B	Z,B,B
	VS	VS,B,S	VS,B,M	VS,B,M	Z,M,M	Z,M,B
	S	S,M,VS	S,M,VS	S,M,VS	VS,S,S	VS,S,S
	M	M,Z,Z	M,Z,Z	M,VS,VS	S,VS,VS	S,VS,VS
	B	B,Z,Z	B,Z,Z	B,Z,Z	B,Z,Z	M,Z,Z

The controllers were built in the Simulink environment combined with Real-time Windows Target toolbox. The sampling rate was set to 1ms. In addition, to evaluate the system stability, a noise source (N) and a disturbance source (D) were generated and in turn added to the controllers' outputs and the Hall sensor feedback signal (Fig. 1) as

$$N(t) = k_N[A_N \sin(2\pi f_N t) + \mathrm{Rand}_N(t)]$$

here A_N was given randomly from -1 to 1; f_N was varied from 1 to 5 Hz; Rand_N is the noise signal with power 0.5; $k_N = 0.02$; and:

$$D(t) = k_D[A_D \sin(2\pi f_D t) + x_D(t)]$$

where

$$x_D(t) = \begin{cases} 1 \;\; IF: & 0 \le (t \bmod T_D) < p_D T_D \\ 0 \;\; IF: & p_D T_D \le (t \bmod T_D) < T_D \end{cases}, \; 0 < p_D < 1$$

here A_D was the Gaussian distribution with mean 0.01 and variance 0.1; f_D was varied from 1 to 0.1 Hz; x_D is the pulse wave signal with the pulse width, p_D, set to 30% of the signal period T_D = 2 s; and k_D = 0.02.

The control target was to drive the magnet to follow a given trajectory defined as the distance from the coil to the magnet. The best working position of the magnet was around a distance of 20mm from the end surface of the driving coil. The real-time tracking control tests on the EMLS were then carried out to validate the applicability of the comparative approaches.

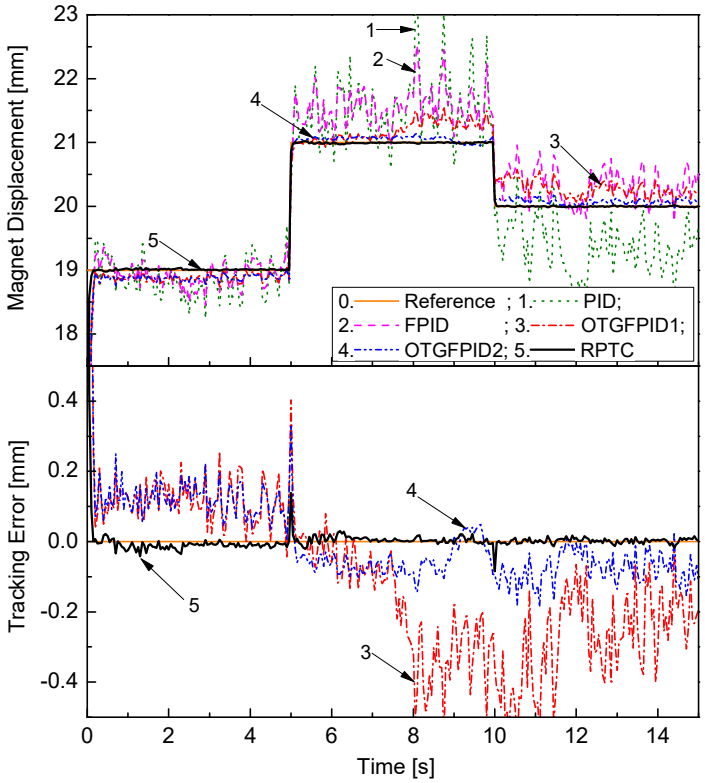


Fig. 10. EMLS multi-step tracking performances with different controllers.

Firstly, the desired trajectory was selected as a multi-step trajectory (Fig. 10 with the top sub-plot). The EMLS was then tested with the five controllers to track the given target. Subsequently, the control performances and tracking errors were obtained and analysed in Fig. 10. From the top sub-plot, it is clearly that both the PID and FPID could not enhance the desired performance

(marked as line number 1 and 2, respectively). Although, the PID gains were tuned based on the optimized system model, the fixed gain use could not keep the system stably when facing with the actual high nonlinear system including the large perturbations. The FPID with the PID gain regulation function could improve the performance of the typical PID. However, it still lacked the adaptability to the noises and disturbances. The control errors in these cases were very large (out of the plotting range in the bottom sub-plot of Fig. 10). With the use of online tuning grey predictor to produce the control action in advance, the OTGFPID1 could provide the better tracking result. However due to the lack of fuzzy parameter tuning function and the limitations of typical grey model (GM(1,1)), the steady state error (*SSE*) was still large as 25% (considering the minimum distance was at 18mm). The performance was then upgraded by the OTGFPID2 which possessed both the adaptive SAGUIGM predictor (Section 5.1a) and the online tuning controller. Although employing the SAUIGM-based predictor with higher accuracy compared to the conventional grey model, there was no constraint to guarantee the robust prediction. Furthermore, the controller lacked of the robust learning and disturbance rejection capability. There reasons led to the limited improvement of the overall tracking result (*SSE* was around 10% in this case). Meanwhile with the use of RPTC approach, in which both the RFPID and RPIGM modules could be robustly optimized in advance, the bad-effects of the generated noises and disturbances on the tracking performance were efficiently eliminated. Since, it could adapt well with the system changes as well as the disturbed environment and, subsequently, the best performance with fast setting time and small *SEE*, within 2%, was obtained (the solid-black lines numbered as 5 in Fig. 10).

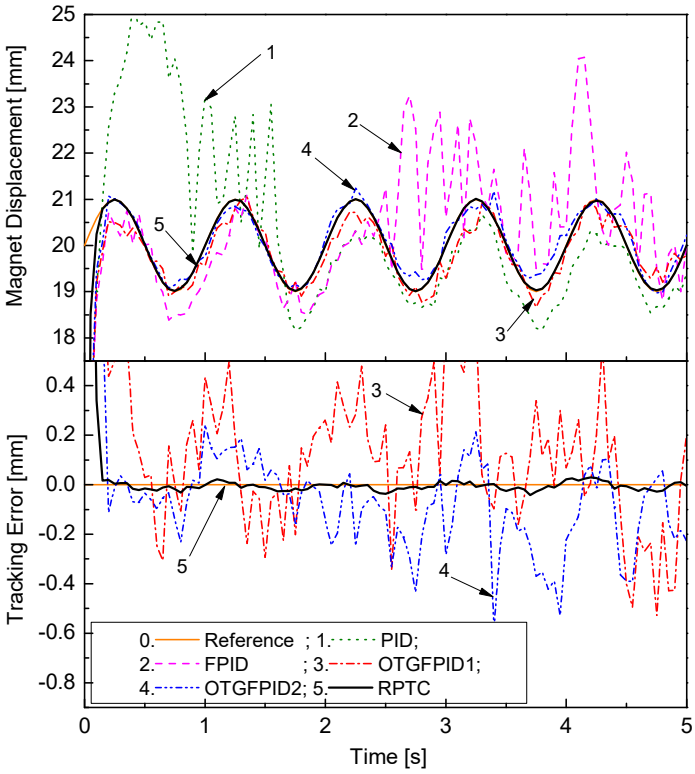


Fig. 11. EMLS sinusoidal tracking performances with different controllers.

Next, the experiments on the EMLS using the compared controllers were performed in which the desired trajectory was changed to a sinusoidal trajectory with 1Hz frequency and 1mm the amplitude around the idea working distance of the magnetic ball (20mm). The control results are then shown in Fig. 11. Similar to the multi-step tracking experiments, the PID controller could not drive the ball to follow the target profile, especially during the first two seconds. When starting, the large system offset with noises and disturbances caused the significant tracking error (the dot-green line (1) of the top sub-plot). This error was slowly lessened but still at the high level. With the FPID, the system could follow the reference with much smaller error compared to the PID case. However after a few seconds, the system performance was diverged and became unstable. The reason was that the FPID without adaptability to noises and disturbances could made the wrong regulation of PID gains and, therefore, the control error was amplified. The performance had better improvements by applying the other three controllers, OTGFPID1, OTGFPID2 and RPTC. Nevertheless, without the robust learning functions of either the fuzzy tuners or the grey predictors and without the noise-disturbance rejection, the tracking errors could not converge to the acceptable range (the errors were around 0.4 to 0.5mm as shown in the bottom sub-plot of Fig. 11). Meanwhile, the RPTC, which contained both the advanced control and prediction functions with robust

constraints, could drive the system to the given target quickly and accurately. The smallest *SSE* (about 3%) was achieved and guaranteed in this case.

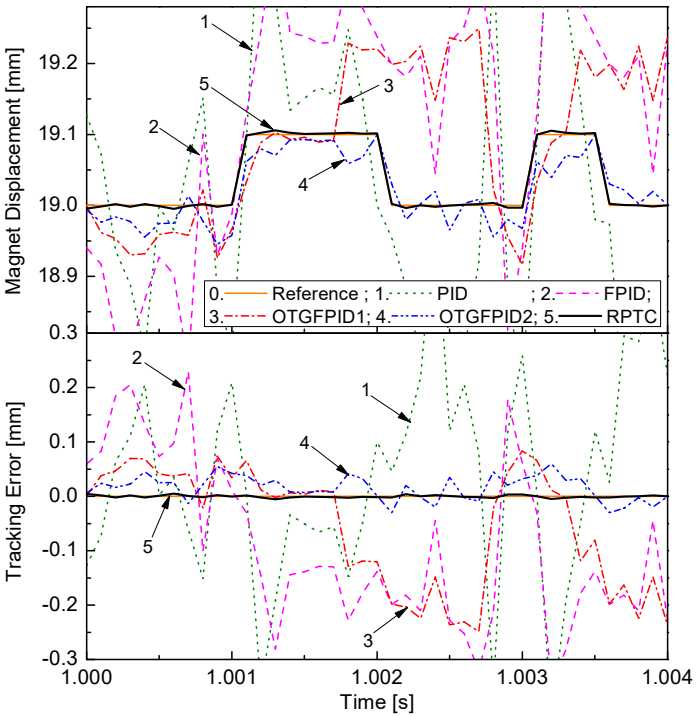


Fig. 12. EMLS high speed tracking performances with different controllers.

Finally, the system was investigated in the high frequency region. The magnet trajectory was selected as a square signal of which the amplitude was 0.1mm and the frequency was at high values: 0.5kHz and 1kHz (see Fig. 12). In this case, the sampling time was set to 10kHz to ensure the system operation and the magnet was driven stably to the working point before starting to track the desired trajectory. In order to perform this experiment series with the high speed sampling rate, the NI multi-function card (PCI-6251 in Section 5.1 and Fig. 8a) has been selected instead of the Hilink board to perform the communication between the EMLS and the controllers at the PC. The experimental results with the different controllers were obtained as plotted in Fig. 12. The results show that the controllers except the RPTC could not drive the highly nonlinear system as the EMLS in the high frequency region. Only using the RPTC, the acceptable performance could be achieved and the error was converged into the acceptable region. It comes as no surprise, since the propose scheme possess: the robust-adaptive control module, RFPID, the robust prediction module, RPIGM, and the ability to compensate the influences of the noises and disturbances on the system performance based on the noise-disturbance estimation.

In summary, the results indicated that the proposed control approach has strong applicability to nonlinear control systems with fast response, high adaptability and stability even in case the working conditions contain large and sudden perturbations.

6. Conclusions

This paper presents the novel control approach – robust predictive tracking control for applications to systems containing large nonlinearities and uncertainties. The RPTC controller is designed with the two advanced technologies: robust fuzzy PID-based control and robust PI grey model-based prediction to ensure the robust tracking performance.

Through the numerical experiments to evaluate the prediction accuracy and stability, it is clear that the RPIGM possess enough ability for both signals forecasting and/or control applications. Next, the effectiveness of the overall control scheme, RPTC, has been validated through the comparative study with the other four controllers in driving the magnet of the EMLS to its desired trajectories. The experimental evaluation proved evidently the advantages of the RPTC over the other methods to guarantee the good tracking with fast response, high stability and disturbance rejection ability even working in the environment containing large perturbations.

Although the accurate control results were ensured by employing the proposed approach, it is realized that the tracking errors was only converged into the acceptable region (*SSE* within 5%) and kept fluctuating within this range (for instance, Fig. 11) because the effects of noises and disturbances have not been fully compensated. The further improvement of this control technique therefore would be a modification in generating the noise-disturbance compensation control signal by using an integral function. Other future research topics would be the development of this approach for sensorless control systems as well as industrial applications.

Acknowledgements

This research is supported by the WMG Centre High Value Manufacturing (HVM) – University of Warwick, in collaboration with the “Next-generation construction machinery component

specialization complex development program" through the Ministry of Trade, Industry and Energy (MOTIE) and Korea Institute for Advancement of Technology (KIAT).

References

[1] P. Hušek, K. Narenathreyas, Aircraft longitudinal motion control based on Takagi–Sugeno fuzzy model, Appl. Soft Comput. 49 (2016) 269–278.
[2] K.K. Ahn, D.Q. Truong, Y.H. Soo, "Self tuning fuzzy PID control for hydraulic load simulator," in Proc. IEEE Int. Conf. Cont., Autom. Syst., Korea, 2007, pp. 345-349.
[3] R. Shanmugasundram, K. Muhammad Zakariah, and N. Yadaiah, Implementation and Performance Analysis of Digital Controllers for Brushless DC Motor Drives, IEEE/ASME Trans. Mechatronics 19 (2) (2014) 213-224.
[4] Y.Z. Li, K.M. Lee, Thermohydraulic Dynamics and Fuzzy Coordination Control of a Microchannel Cooling Network for Space Electronics, IEEE Trans. Ind. Electron. 58 (2) (2011) 700-708.
[5] M.M. Rashid, N.A. Rahim, M.A. Hussain, M.A. Rahman, Analysis and Experimental Study of Magnetorheological-Based Damper for Semiactive Suspension System Using Fuzzy Hybrids, IEEE Trans. Ind. Applic. 47 (2) (2011) 1051-1059.
[6] W.D. Chang, R.C. Hwang, J.G. Hsieh, A self-tuning PID control for a class of nonlinear systems based on the Lyapunov approach, J. Process Control 12 (2002) 233–242.
[7] Y.T. Juang, Y.T. Chang, C.P. Huang, Design of fuzzy PID controllers using modified triangular membership functions, Inf. Sci. 178 (2008) 1325-1333.
[8] S.C. Wang, Y.H. Liu, A Modified PI-Like Fuzzy Logic Controller for Switched Reluctance Motor Drives, IEEE Trans. Ind. Electron. 58 (5) (2011) 1812-1825.
[9] K.K. Ahn, D.Q. Truong, T.Q. Thanh, B.R. Lee, Online self-tuning fuzzy proportional–integral– derivative control for hydraulic load simulator, Proc. IMechE Part I: J. Syst. Cont. Eng. 222 (2) (2008) 81-95.
[10] D.Q. Truong, K.K. Ahn, Self Tuning of Quantitative Feedback Theory for Force Control of an Electro-Hydraulic Test Machine, Cont. Eng. Prac. 17 (11) (2009) 1291-1306.
[11] J.L. Meza, V. Santibáñez, R. Soto, M.A. Llama, Fuzzy Self-Tuning PID Semiglobal Regulator for Robot Manipulators, IEEE Trans. Ind. Electron., 59 (6) (2012) 2709-2717.
[12] X.G. Duan, H. Deng, H.X. Li, A Saturation-based Tuning Method for Fuzzy PID Controller, IEEE Trans. Ind. Electron. 60 (11) (2013) 5177-5185.
[13] Y. Li, C. Cai, K.-M. Lee, F. Teng, A Novel Cascade Temperature Control System for a High-Speed Heat-Airflow Wind Tunnel, IEEE/ASME Trans. Mechatronics 18 (4) (2013) 1310-1319.
[14] I. Siradjuddin, L. Behera, T.M. McGinnity, S. Coleman, Image-Based Visual Servoing of a 7-DOF Robot Manipulator Using an Adaptive Distributed Fuzzy PD Controller, IEEE/ASME Trans. Mechatronics 19 (2) (2014) 703-710.
[15] K. Erenturk, Hybrid Control of a Mechatronic System: Fuzzy Logic and Grey System Modeling Approach, IEEE/ASME Trans. Mechatronics 12 (6) (2007) 703-710.
[16] D.Q. Truong, K.K. Ahn, Force control for hydraulic load simulator using self-tuning grey predictor – fuzzy PID, Mechatronics 19 (2) (2009) 233-246.

[17] C.M. Druitt, G. Alici, Intelligent Control of Electroactive Polymer Actuators Based on Fuzzy and Neurofuzzy Methodologies, IEEE/ASME Trans. Mechatronics 19 (6) (2014) 1951-1962.
[18] I. Chairez, Differential Neuro-Fuzzy Controller for Uncertain Nonlinear Systems, IEEE Trans. Fuzzy Syst. 21 (2) (2013) 369-384.
[19] M.B.K. Bouzid, G. Champenois, N.M. Bellaaj, L. Signac, K. Jelassi, An Effective Neural Approach for the Automatic Location of Stator Interturn Faults in Induction Motor, IEEE Trans. Ind. Electron. 55 (12) (2008) 4277-4289.
[20] F. Moreno, J. Alarcón, R. Salvador, T. Riesgo, Reconfigurable Hardware Architecture of a Shape Recognition System Based on Specialized Tiny Neural Networks With Online Training, IEEE Trans. Fuzzy Syst. 56 (8) (2009) 3253-3263.
[21] C. Xia, C. Guo, T. Shi, A Neural-Network-Identifier and Fuzzy-Controller-Based Algorithm for Dynamic Decoupling Control of Permanent-Magnet Spherical Motor, IEEE Trans. Ind. Electron. 57 (8) (2010) 2868-2878.
[22] C.H. Wang, D.Y. Huang, A New Intelligent Fuzzy Controller for Nonlinear Hysteretic Electronic Throttle in Modern Intelligent Automobiles, IEEE Trans. Ind. Electron. 60 (6) (2013) 2332-2345.
[23] S.K. Jain, S.N. Singh, Low-Order Dominant Harmonics Estimation using Adaptive Wavelet Neural Network, IEEE Trans. Ind. Electron. 61 (1) (2014) 428-435.
[24] V.S. Kodogiannis, A. Alshejari, An adaptive neuro-fuzzy identification model for the detection of meat spoilage, Appl. Soft Comput. 23 (2014) 483–497.
[25] J.L. Deng, Introduction to Grey System Theory, J. of Grey Syst. 1 (1989) 1-24.
[26] R.C. Luo, T.M. Chen, Autonomous Mobile Target Tracking System Based on Grey-Fuzzy Control Algorithm, IEEE Trans. Ind. Electron. 47 (4) (2000) 920-931.
[27] R.J. Wai, C.H. Tu, Adaptive Grey Control for Hybrid Resonant Driving Linear Piezoelectric Ceramic Motor, IEEE Trans. Ind. Electron. 53 (2) (2006) 640-656.
[28] R. Guo, Modeling imperfectly repaired system data via grey differential equations with unequal-gapped times, Rel. Eng. & Sys. Safety 92 (3) (2007) 378-391.
[29] L.R. Chen, R.C. Hsu, C.S. Liu, A Design of a Grey-Predicted Li-Ion Battery Charge System, IEEE Trans. Ind. Electron. 55 (10) (2008) 3692-3701.
[30] E. Kayacan, Y. Oniz, O. Kaynak, A Grey System Modeling Approach for Sliding-Mode Control of Antilock Braking System, IEEE Trans. Ind. Electron. 56 (8) (2009) 3244-3252.
[31] E. Kayacan, B. Ulutas, O. Kaynak, Grey system theory-based models in time series prediction, Expert Syst. Appl. 37 (2) (2010) 1784-1789.
[32] G.D. Li, S. Masuda, D. Yamaguchi, M. Nagai, An Optimal Grey PID Control System, IEEJ Trans. Electric. Electron. Eng. 4 (4) (2009) 570–577.
[33] R.J. Lian, Grey-prediction self-organizing fuzzy controller for robotic motion control, Inf. Sci. 202 (20) (2012) 73-89.
[34] L. Wu, S. Liu, Y. Yang, Grey double exponential smoothing model and its application on pig price forecasting in China, Appl. Soft Comput. 39 (2016) 117–123.
[35] D.Q. Truong, K.K. Ahn, N.T. Trung, Design of An Advanced Time Delay Measurement and A Smart Adaptive Unequal Interval Grey Predictor for Real-time Nonlinear Control Systems, IEEE Trans. Ind. Electron. 60 (10) (2013) 4574-4589.
[36] X. Dong, Y. Lian, Y. Liu, Small and multi-peak nonlinear time series forecasting using a hybrid back propagation neural network, Infor. Sci. 424 (2018) 39–54.
[37] A. Madkour, M.A. Hossain, K.P. Dahal, H. Yu, Intelligent Learning Algorithms for Active Vibration Control, IEEE Trans. on Syst., Man, and Cyber. - Part C: App. and Rev., 37 (5) (2007) 1022-1033.

[38] C.D. Stylios, V.C. Georgopoulos, G.A. Malandraki, S.S. Chouliara, Fuzzy cognitive map architectures for medical decision support systems, Appl. Soft Comput. (8) (2008) 1243-1251.
[39] D. Yaman, S. Polat, A fuzzy cognitive map approach for effect-based operations: An illustrative case, Infor. Sci. 179 (2009) 382-403.
[40] G. Kyriakarakos, K. Patlitzianas, M. Damasiotis, D. Papastefanakis, A fuzzy cognitive maps decision support system for renewables local planning, Renew. Sust. Energ. Rev. 39 (2014) 209-222.
[41] P.J. Giabbanellia, T. Torsney-Weira, V.K. Magoa, A fuzzy cognitive map of the psychosocial determinants of obesity, Appl. Soft Comput. 12(12) (2012) 3711–3724.
[42] E.I. Papageorgiou, P.P. Groumpos, A new hybrid method using evolutionary algorithms to train Fuzzy Cognitive Maps, Appl. Soft Comput. 5 (4) (2005) 409–431.
[43] Zeltom Company, Electromagnetic levitation system - User manual. [Online]. Available: www.zeltom.com/documents/emls_um_14.pdf.
[44] Zeltom Company, Hilink system - User manual. [Online]. Available: http://www.zeltom.com/documents/hilink_qr_21.pdf.

Appendix

The Lyapunov function is properly selected as

$$V(t+1)=\frac{1}{2}\left[y_r(t+1)-y(t+1)\right]^2=\frac{1}{2}e_{t+1}^2 \tag{A1}$$

The change of this lyapunov function is derived as

$$\Delta V(t+1)=V_{t+1}-V_t=\frac{1}{2}\left(e_{t+1}^2-e_t^2\right)=e_t\Delta e_t+\frac{1}{2}\Delta e_t^2 \tag{A2}$$

$$e_{t+1}=e_t+\Delta e_t \tag{A3}$$

From the configuration of the RFPID controller (Fig. 2)

$$\Delta e_t=\sum_{i=1}^{2}\sum_{j=1}^{N}\left(\frac{\partial e_t}{\partial a_{ijt}}\Delta a_{ijt}+\frac{\partial e_t}{\partial b_{ijt}^{+/-}}\Delta b_{ijt}^{+/-}\right)+\sum_{l=1}^{M}\frac{\partial e_t}{\partial w_{lt}}\Delta w_{lt} \tag{A4}$$

From Fig. 2, and note that $\partial x_{it}/\partial e_t=\pm k_i$, one has

$$\frac{\partial e_t}{\partial a_{ijt}}=\left(\frac{\partial f_{ij}}{\partial a_{ijt}}\right)/\left(\frac{\partial f_{ij}}{\partial e_t}\right)=\left(\mp\frac{1}{b_{ijt}^{+/-}}\right)/\left(\pm\frac{(\pm k_i)}{b_{ijt}^{+/-}}\right)=\pm k_i \tag{A5}$$

Similarly,

$$\frac{\partial e_t}{b_{ijt}^{+/-}} = \left(\frac{\partial f_{ij}}{b_{ijt}^{+/-}}\right) / \left(\frac{\partial f_{ij}}{\partial e_t}\right) = \pm \frac{\left(x_{it} - a_{ijt}\right)}{k_i b_{ijt}^{+/-}} \tag{A6}$$

$$\frac{\partial e_t}{\partial w_{lt}} = \left(\frac{\partial U_{FAt}}{\partial w_{lt}}\right) / \left(\frac{\partial U_{FAt}}{\partial e_t}\right) \equiv F_1 \tag{A7}$$

Here $\partial U_{FAt} / \partial e_t$ is computed similar as calculating $\partial U_{FAt} / \partial a_{ijt}$. The terms $\Delta a_{ijt}, \Delta b_{ijt}^{+/-}, \Delta w_{lt}$ are calculated by (19)-(29). From (18), (A5)-(A7), and by selecting $\eta_{at} = \eta_{bt}$, (A4) becomes:

$$\Delta e_t = \eta_{at} \sum_{i=1}^{2} \sum_{j=1}^{N} \left(\pm k_i \frac{\partial E_{t+p}}{\partial a_{ijt}} \pm \frac{\left(x_{it} - a_{ijt}\right)}{k_i b_{ijt}^{+/-}} \frac{\partial E_{t+p}}{\partial b_{ijt}^{+/-}} \right) + \eta_{wt} \sum_{l=1}^{M} F_1 \frac{\partial E_{t+p}}{\partial w_{lt}} \tag{A8}$$

From (A8), (A2) is represented as

$$\Delta V\left(t+1\right) = e_t \left(\eta_{at} F_2 + \eta_{wt} \sum_{l=1}^{M} F_1 \frac{\partial E_{t+p}}{\partial w_{lt}} \right) + \left(\eta_{at} F_2 + \eta_{wt} \sum_{l=1}^{M} F_1 \frac{\partial E_{t+p}}{\partial w_{lt}} \right)^2 / 2 \tag{A9}$$

The RPTC control system is guaranteed to be stable only if $\Delta V\left(t+1\right) \le 0$ for all steps. According to (A9), it is clear that except η_{at} and η_{wt}, the other factors are determinable for each working step. Since, by selecting these learning rates to satisfy (30), the sufficient condition for $\Delta V\left(t+1\right) \le 0$ is achieved.